Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Igor Plavšić

Mobilna aplikacija za odčitavanje in ocenjevanje izdelkov

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Rok Rupnik

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Zasnujte in razvijte mobilno aplikacijo potrošnike, ki želijo v trgovinah odčitavati črtne kode izdelkov ter na podlagi tega bodisi pregledovati komentarje ostalih potrošnikov, bodisi vnašati svoje komentarje za izdelke. Mobilno aplikacijo razvijte za Android platformo, za shranjevanje podatkov pa uporabite MySQL podatkovno bazo.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Igor Plavšić, z vpisno številko **63050339**, sem avtor diplomskega dela z naslovom:

Mobilna aplikacija za odčitavanje in ocenjevanje izdelkov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Roka Rupnika,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 16. septembra 2014 Podpis avtorja:

Hvala moji družini in punci, ki so me spodbujali ter mi stali ob strani v času študija. Zahvalil bi se tudi mentorju doc. dr. Roku Rupniku za vso pomoč pri izdelavi diplomskega dela.

Kazalo

Povzetek

V diplomskem delu je opisan postopek izdelave mobilne aplikacije za Android platformo. Aplikacija omogoča odčitavanje črtne kode izdelkov. Če izdelka v bazi ni, ga kreira, v nasprotnem primeru pa omogoči ocenjevanje oz. komentiranje na tega. Aplikacija uporabi kamero telefona za zajem slike črtne kode izdelka. Sliko aplikacija analizira s pomočjo Googleovega QR čitalca (*Barcode Scanner*), ki je integriran vanjo, ta pa kot rezultat operacije vrne pripadajočo šifro črtne kode. Izdelke se shranjuje na spletni strežnik preko HTTP poizvedb v MySQL podatkovno bazo. Osrednja tema diplomskega dela je delovanje QR čitalca, ki zajame in analizira črtno kodo, kot tudi implementacija komunikacije s spletnim strežnikom.

Ključne besede: Android, QR čitalec, komunikacija s spletnim strežnikom.

Abstract

The thesis describes the development procedure of a mobile application for the Android platform. The application enables you to scan barcodes of products. If a product exist in the database, the application enables the user to rate and comment the product, if not the application creates a new entry into the product database. The application uses the camera on the phone to capture an image of the products barcode, which is then analysed by the applications integrated Googles Barcode Scanner. The result of the analysis is a numeric representation of the barcode. All data is stored in a MySQL database on a web server with which we communicate by HTTP request. THe main topic of the thesis is to explain how the QR scanner wors aswell as how the communication with the web server.

Keywords: Android, QR Scanner, Web Server communication.

Poglavje 1

Uvod

Mobilna aplikacija, razvita v sklopu diplomskega dela, omogoča uporabniku, da opravi poizvedbo o izbranem izdelku ali vnos novega v sistem. Aplikacija omogoča tudi ocenjevanje in komentiranje na izbrane izdelke, zato bi lahko rekli, da se jo lahko uporablja kot neke vrste ocenjevalca kakovosti izdelkov, pred nakupom tega, ki je izključno vodeno s strani uporabnikov. Aplikacija je lahko uporabljena tudi kot alternativa brskanju po internetu za ocene in mnenja uporabnikov na spletnih straneh in forumih. Tako lahko uporabnik, ki je v dilemi glede nakupa izdelka, ki ga drži v roki, takoj vidi ocene izdelka, ki so jih pustili drugi uporabniki aplikacije, in se odloči ali premisli glede nakupa izdelka.

Mobilna aplikacija je razvita na platformi `Android`, razlog za to pa je njegova odprtost in razširjenost v svetu mobilnega računalništva. V kratkem času njegovega obstoja je prevzel dominantno vlogo in predstavlja logično odločitev za doseganje čim večje ciljne publike uporabnikov. Znotraj aplikacije je uporabljenih več tehnologij, med glavni dve štejeta odprtokodni projet `ZXing`, ki se ukvarja z računalnikim vidom, in sicer procesiranjem črtnih in QR kod, ter komunikacija aplikacije z oddaljeno podatkovno bazo preko spletnih poizvedb.

Prvi del diplomskega dela vsebuje predstavitev
tt Android platforme ter njegovih razvojnih orodij, ki so bili uporabljeni za

razvoj aplikacije, ter opis uporabljenih tehnologij, ki se jih aplikacija poslužuje. V drugem delu pa bo predstavljen razvojni postopek in glavne ovire razvoja mobilne aplikacije.

Poglavje 2

Android

Android je operacijski sistem, ki se ga uporablja primarno v mobilni industriji kot tudi za tablične računalnike, pred kratkim pa se je v okrnjeni verziji začel uporabljati tudi za nosljive naprave (angl. *Wearable devices*). Gonilna sila razvoja Androida je Google, ki ga izdaja pod odprtokodnimi licencami. Ker je Android odprtokodna rešitev, pomeni, da ga lahko podjetja ali posamezniki prikrojijo za svoje potrebe, kar je tudi eden izmed glavnih razlogov, zakaj je Android trenutno vodilna mobilna platforma glede na število aktivnih uporabnikov.

2.1 Zgodovina

Začetki Android platforme segajo v leto 2007, ko ga je Google skupaj s konzorcijem drugih podjetij predstavil, s končnim namenom, da postavijo odprte standarde za razvoj mobilnih naprav. Trenutna verzija Androida je 4.4.4. bolje znana pod svojim aliasom KitKat, napovedana pa je tudi nova verzija, katere alias je Android L. Z izidom zadnje verzije bo to 20 instanca Android platforme. Celotno zgodivino verzij si lahko preberete v tabeli 2.1 [1].

Verzija	Alias	API
1.0	/	1
1.1	/	2
1.5	Cupcake	3
1.6	Donut	4
2.0 - 2.1	Eclair	5 - 7
2.2.*	Froyo	8
2.3.*	Gingerbread	9 - 10
3.*	Honeycomb	11 - 13
4.0.*	Ice Cream Sandwich	14 - 15
4.1. - 4.3.	Jeally Bean	16 - 18
4.4.*	KitKat	19 - 20
5.0	Android Lolipop	21

Tabela 2.1: Zgodovina verzij platforme Android

2.2 Razšijenost platforme in njena predvidena prihodnost

V kratkem času svojega obstoja se je Android platforma hitro povzpela na prvo mesto platform za pametne mobilne naprave in tablične računalnike. Po podatkih, ki jih vodi Google, naj bi se vsak dan aktiviralo več kot milijon novih naprav. Čeprav je Android platforma, in posledično Google, na prvem mestu, ta ne počiva na svojih lovorikah ter že gledajo, kako bi platformo lahko razširili na več naprav. Letos (2014) so že predstavili razširitev na pametne ure, ki bi delovale na okrnjeni verziji platforme, imenovana `Andorid Wear`, razvoj pa poteka tudi za pametne televizije (`Andorid TV`) ter celo avtomobile (`Andorid Auto`) [2].

2.3 Razvoj aplikacij na Android platformi

2.3.1 Android SDK

Android SDK (`angl. Software Development Kit` - paket za razvoj programske opreme) je nabor orodij in knjižnic, ki prevedejo kodo, napisano v programskem jeziku `Java` ter jo zapakira v izvedljivo datoteko, ki jo uporabimo za namestitev aplikacije na mobilno napravo, ki jo poganja Android platforma. Orodja vključujejo tudi razne API-je (angl. *Application Programming Interface* - programski vmesnik) za `Google`ove storitve, kot so `Google Play`, `Google Maps`,... [3]

2.3.2 Eclipse ADT

Eclipse ADT (angl. `Android Developer Tools` - razvojna orodja za Android) je integrirano razvojno okolje Eclipse z nameščenim vtičnikom ADT. Skupaj postaneta močno orodje za razvoj Android aplikacij. Eclipse s pomočjo SDK-ja omogoča ustvarjanje novih kot tudi razhroščevanje obstoječih aplikacij. Pohitri kreiranje hiearhije projekta s posebnimi čarovniki ter poenostavi razvoj uporabniških vmesnikov z vizualnim oblikovalnim orodjem [4].

2.3.3 Emulator naprav

Eno od orodij, ki jih dobimo skupaj z SDK-jem, je tudi emulator, ki emulira napravo s poljubno konfiguracijo(*dimenzije ekrana, procesorska moč, ...*) kot tudi poljubno verzijo Android operacijskega sistema. Tako omogoča razvijalcu, da razvija in testira svoje aplikacije za več različnih konfiguracijah naprav, brez da bi si katero fizično lastil [4].

Poglavje 3

Uporabljene tehnologije pri izdelavi aplikacije

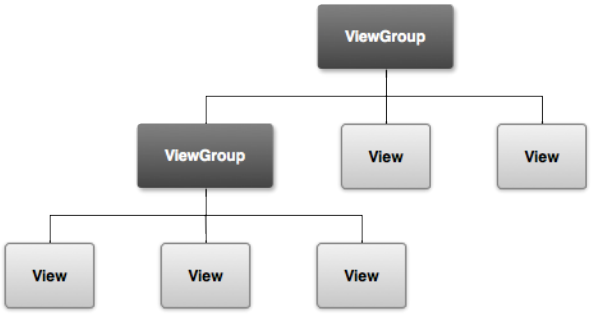
Za razvoj aplikacije so bile uporabljene razne tehnologije, ki so bile potrebne za njeno uspešno implementacijo. Tehnično bolj podrobne opise pa si lahko preberete v Poglavju 4.

3.1 Komponente Android SDK-ja

Aplikacija brez Androidovih komponent ne more zaživeti na Android platformi. V tem razdelku si bomo ogledali komponente, ki so nujne za pravilno delovanje katerekoli Andorid aplikacije, kot so *Aktivnosti* (angl. *Activity*) in sestavo uporabniškega vmesnika ter kako Andorid shranjuje podatke, vezane na aplikacijo, v aplikativni spomin [3].

3.1.1 Pogled in skupina pogledov

Pogled in skupine pogledov (angl. *View* in *ViewGroup*) sta komponenti, iz katerih so sestavljeni vsi elementi uporabniškega vmesnika (*gumb, vnosno polje, seznam, ...*). Pogled je komponenta, ki element izriše na zaslon, kar omogoča uporabniško interakcijo. Vsaka komponenta aplikacije je definirana kot hiearhija pogledov in skupin pogledov (Slika 3.1). Pogled lahko defi-



Slika 3.1: Hiearhija pogleda in skupin pogledov

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="I am a Button" />
</LinearLayout>
```

Slika 3.2: Primer XML-a za pogled

niramo tako, da dedujemo od razreda `View` ali `ViewGroup` in znotraj njega definiramo pozicije komponent. Obstaja pa tudi lažji in bolj intuitiven način, in sicer v XML (angl. *EXtensible Markup Language*) datoteki, kjer je hierhija pogleda samoumevna glede na sestavo XML-a (Slika 3.2) [5] [6].

3.1.2 Aktivnost

Aktivnost (angl. *Activity*) je kompomenta, na katero je vezana ena maska uporabniškega vmesnika. Kar pomeni, da je vsaka aktivnost nov ekran, preko katerega uporabnik manipulira z aplikacijo in podatki znotraj te. Aplikacije so ponavadi sestavljene iz večih aktivnosti, ki so ohlapno povezane med seboj.

V večini aplikacij je ena izmed teh aktinosti označena kot glavna (angl. *main*) ter se prikaže, ko jo uporabnik odpre. Vsaka aktivnost lahko zažene drugo. Ob zagonu nove aktivnosti se stara ustavi, Android pa shrani njeno stanje v sklad predhodnih aktivnosti (angl. *back stack*). Kar pomeni, da če uporabnik pritisne gumb za nazaj, se trenutna aktivnost uniči, predhodna pa povleče ven iz sklada in restavrira na stanje pred prehodom v novo aktivnost.

Da se lahko kreira aktivnost, moramo dedovati od razreda `Activity` in implementirati metode povratnih klicev (angl. *callback methods*), ki jih sistem kliče med izvajanjem aplikacije. Najpomembnejši dve metodi sta `onCreate()` in `onPause()`. Sistem prvo metodo kliče ob kreiranje instance aktivnosti, v njej pa ponavadi nastavimo vse, kar je pomembno za pravilno delovanje aktivnosti, obvezna pa je nastavitev razporeditve uporabniškega vmesnika (angl. *layout*) s klicem metode `setContentView()`. Druga se izvede ob prvih znakih, da se bo aktivnost uničila, v njej pa ponavadi poskrbimo za zadnje potrditve morebitnih transakcij, ki se izvajajo [7].

3.1.3 Nameni

Nameni (angl. *Intents*) so komponente, preko katerih druge komponente (*npr. Activity*) pošiljajo zahteve drugim komponentam za izvedbo nekega zaporedja dogodkov. Namene se primarno uporablja za zagon aktivnosti, ki ga izvedemo z ukazom `startActivity()` ali `startActivityForResult()`, če želimo pridobiti povratne informacije iz zagnane aktivnosti. Uporablja pa se jih tudi za zagon storitev (angl. *Service*), ki se izvajajo v ozadju in nimajo uporabniškega vmesnika, ter za hkratno oddajanje (angl. *Broadcast*) sporočil določenim aplikacijam o sistemskih dogodkih [8].

3.1.4 Asinhrona naloga

Asinhrone naloge (angl. *AsyncTask*) omogočajo enostavno uporabo niti uporabniškega vmesnika. Asinhrone naloge se uporablja za kratkotrajne zaledne opracije, katere razultat je vrnjen na nit uporabniškega vmesnika [25].

3.1.5 Upravljalec nastavitev

Upravljalec nastavitev (angl. *PreferenceManager*) je orodje znotraj Android SDK-ja, ki omogoča shranjevanje in opravljanje s podatki, vezanimi na aplikacijo in njenimi nastavitvami [9].

3.2 Čitalec črtnih kod

Čitalec črtnih kod (angl. *Barcode Scanner*) je Googleova odprtokodna aplikacija za branje eno- in dvodimenzionalnih QR kod (angl. *Quick Response Code*), ki uporablja ZXing knjižnice za analizo teh. Aplikacija analizira sliko kode in vrne podatke, ki so lahko numerične ali tekstovne šifre, spletni naslovi, virtualne vizitke oseb, binarna kodirana vsebina in še marsikaj drugega. Pri implementaciji mobilne aplikacije sem se odločil, da bo čitalec vgrajen v aplikacijo in se tako rešil problema, če slučajno uporabnik ne bi imel nameščene aplikacije, ki omogoča branje črtnih kod [11].

3.2.1 Črtne in QR kode

Črtne kode so optične, eno-dimenzionalne, strojno berljive predstavitve podakov, ki so odvisni od razmikov med, in širini, navpičnih črt. Podatki so zaradi enostavnosti zgradbe zapisa omejene na le nekaj znakov dolžine. Nadgradnja navadne črtne kode so QR kode, ki so sestavljene iz več črnih kvadratkov različnih velikosti. QR kode so bolj popularne ravno zaradi prostorke nadgradnje, saj je količina podatkov, ki jih lahko shranimo v QR kodo, neprimerljivo večja kot z navadno črtno kodo. Primera obeh vrst kod prikazuje Slika 3.3 [10].

3.2.2 ZXing projekt

Je odprtokodni projekt za analizo in branje QR kod, implementiran v programskem jeziku `Java`, ki je prenesen na več drugih platform. Google knjižnico uporablja v svoji aplikacije `Barcode Scanner`. Da sprožimo branje črtne ali

Slika 3.3: Primer črtne in QR kode

QR kode s čitalnikom iz druge aplikacije uporabimo namen (angl. *Intent*), ki nato posreduje rezultat branja nazaj klicujoči aplikaciji. Ta rešitev ima eno pomankljivost; to je, da se zanašamo na to, da imamo čitalec že nameščen na napravi. To pomankljivost rešimo tako, da vgradimo čitalec znotraj aplikacije (postopek opisan v 4.6) [11].

3.3 Spletni strežnik in podatkovna baza

Da je katerakoli aplikacija klasificirana kot večuporabniška, se morajo podatki, s katerimi upravlja, shranjevati na oddaljeni podatkovni bazi, ki je s to aplikacijo dostopna vsem napravam. Pri implementaciji aplikacije sem za potrebe stežnika uporabljal program `XAMPP` (več v nadaljevanju). Google že ima rešitve, ki vključujejo vsa ta orodja (*Google App Engine*), a jaz sem hotel ostati fleksibilen za morebitne nadgradnje projekta, kot so izdelava spletne strani, ki uporablja iste skripte za dostop do podatkov.

3.3.1 XAMPP

XAMPP je odprtokodno razvojno orodje, ki omogoča testiranje spletnih aplikacij, saj vsebuje vse potrebno za emulacijo spletnega strežnika. Skupaj z namestitvenim paketom je vključen `MySQL` sistem za upravljanje s podatkovnimi bazami ter interpreter za `PHP` in `Perl` skriptna jezika. Z malo dodatne konfiguracije se XAMPP lahko uporabi kot produkcijsko orodje za upravljanje spletnega strežnika in ne samo kot razvojno orodje. [12]

Apache spletni stežnik

Skupaj z XAMPP razvojim orodjem dobimo množico orodji, ki pripomorejo pri razvoju spletnih aplikacij. Najpomembnejši od teh je Apache HTTP spletni strežnik. Apache HTTP projekt je odprtokodni projekt, katerega cilj je razvoj robustnega spletnega strežnika, ki bi omogočal zanesljivo komunikacijo preko spleta.

Naloga spletnega strežnika je, da omogoča komunikacijo med uporabnikom in spletno aplikacijo, ki se izvaja na strežniku. Da strežniki lahko to nalogo opravljajo, so opremljeni s prevajalniki, ki omogočajo izvajanje programske kode spletnih aplikacij. Med te spadajo prevajalniki za PHP, Perl, Python in druge [13].

3.3.2 MySQL sistem za upravljanje podatkovnih baz

MySQL je odprtokodni sistem za upravljanje relacijskih podatkovnih baz in je najbolj razširjen pri implementaciji spletnih aplikacij. Orodje omogoča načrtovanje in urejanje sestave podatkovne baze, kot je kreiranja in urejanje tabel podatkovne baze in urejanje zapisov znotraj njih, kreiranje procedur ali funkcij za urejanje podatkov kot tudi določanje varnostnih pravil, kako se uporabniki lahko povezujejo do baze. Dostop do podatkov je omogočen preko SQL (angl. *Structured Query Language*) skriptnega jezika (več v 4.3.3) [14].

Relacijska podatkovna baza

Je baza, ki shrani podatke in prikaže tudi, kako so podatki povezani (v kakšnih relacijah so) med seboj. Vsaka relacijska podatkovna baza je zbirka tabel, ki so med seboj povezane glede na medsebojne odvisnosti. Vsaka tabela ima svoj ključ preko katerega je zagotovljeno, da se zapisi v tabeli ne podvajajo. Povezavo z drugo podatkovno tabelo dosežemo tako, da v tabeli definiramo polje, tuj ključ, ki je ključ druge tabele [15].

Poglavje 4

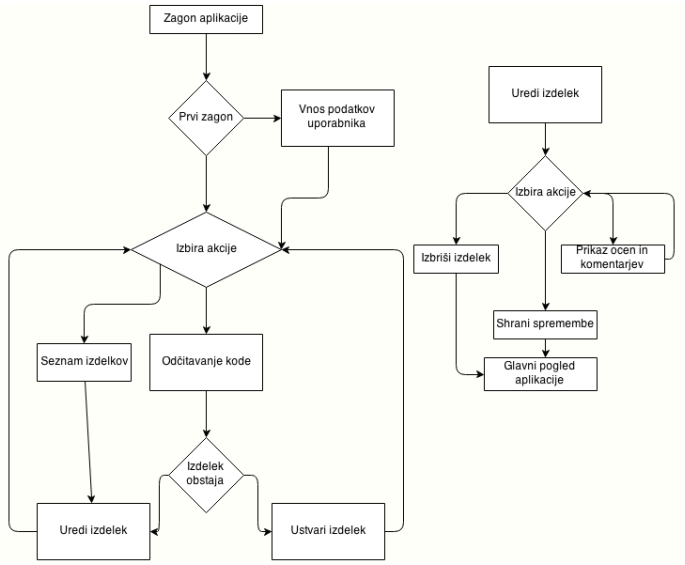
Razvoj aplikacije

V temu poglavju bo predstavljen razvojni proces mobilne aplikacije. Najprej bo sledil opis diagrama toka dogodkov aplikacije ter nivojski opis arhitekture aplikacije. Arhitektura ima tri nivoje ... podatkovna baza, spletni strežnik ter aplikacija oz. uporabniški vmesnik. Opis razvojnega procesa bo potekal po arhitekturi navzgor, se pravi, da bomo začeli s strukturo podatkovne baze, nato skriptiranja komunikacije s podatkovno bazo na spletnem strežniku, za tem se premaknemo na aplikacijo samo, kjer bomo pregledali, kako aplikacija komunicira s strežnikom, za konec pa bomo predstavili razvoj uporabniškega vmesnika ter integracijo čitalca črtnih kod.

4.1 Diagram toka dogodkov

Diagram predstavlja zaporedje dogodkov, ki se zgodijo ob uporabi aplikacije. Ob prvem vstopu v aplikacijo se prikaže pogled za vnos uporabnika. Po uspešnem vnosu podatkov uporabnika se prikaže glavni pogled, kjer izberemo med prikazom seznama izdelkov, že shranjenih v sistemu, in odčitavanjem novega. Če se odločimo za slednjo, se zažene odčitavanje črtne kode, katere rezultat se uporabi za poizvedbo, ali izdelek že obstaja v sistemu. Če ne obstaja, se zažene pogled za vnos, drugače pa za urejanje izdelka. Pri urejanju podatkov izdelka imamo več možnih akcij. Ena je shranjevanje spremenjenih

podatkov, druga izbris izdelka ter vseh podatkov, povezanih z njim, tretja pa prikaz ocen in komentarjev izbranega izdelka.



Slika 4.1: Diagram toka dogodkov aplikacije

4.2 Podatkovna baza

Podatkovna baza aplikacije, poimenovana `QR_PRODUCTS`, vsebuje tri tabele, kjer se shranjuje vse, kar je pomembno za pravilno delovanje uporabniškega vmesnika na mobilni napravi. Komunikacija s podatkovno bazo poteka izključno preko spletnega strežnika (več v 4.3) in nič preko uporabniškega vmesnika.

4.2.1 Tabele

Podatkovna `QR_PRODUCTS` baza vsebuje tri tabele. Tabele so med seboj povezane in zapisi v njih so med seboj odvisni glede na te povezave. Povezave med tabelami ter strukturo navedenih tabel prikazuje slika 4.2.

PRODUCTS

Tabela vsebuje podatke o izdelkih, ki jih shranimo ali ažuriramo preko delovanja aplikacije.

- `PID` - Ključ izdelka
- `U_ID` - Uporabniško ime uporabnika, ki je kreiral zapis
- `QR_CODE` - Številska predstavitev, odčitane črtne kode izdelka
- `NAME` - Naziv izdelka
- `PRICE` - Cena izdelka
- `DESCRIPTION` - Opis izdelka
- `CREATED_AT` - Časovni žig vnosa zapisa
- `UPDATED_AT` - Časovni žig posodobitve zapisa

USER_TABLE

Tabela, v kateri so shranjeni vsi uporabniki, ki so se vpisali preko aplikacije.

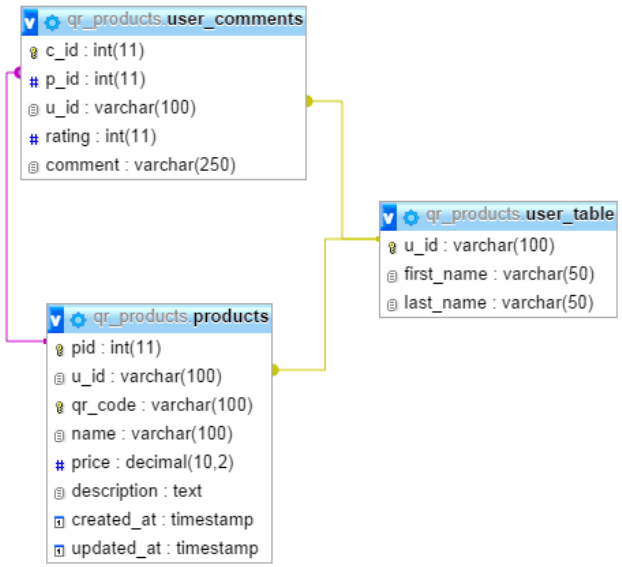
- `U_ID` - Uporabniško ime uporabnika
- `FIRST_NAME` - Ime uporabnika
- `LAST_NAME` - Priimek uporabnika

USER_COMMENTS

Tabela vsebuje komentarje ter ocene izdelkov, ki so jih podali uporabniki.

- `C_ID` - Ključ komentarja
- `PID` - Ključ izdelka, ki ga uporabnik ocenjuje/komentira
- `U_ID` - Uporabniško ime uporabnika, ki je kreiral zapis

- `RATING` - Ocena izdelka
- `COMMENT` - Uporabnikov komentar na izdelek



Slika 4.2: Struktura podatkovne baze QR_PRODUCTS

4.3 Spletni strežnik

Naslednji nivo hiearhije informacijskega sistema je spletni sprežnik, kjer se izvaja vsa poslovna logika spletne aplikacije. Spletni strežnik opravlja vso komunikacijo med uporabniškim vmesnikom (mobilna aplikacija) in podatkovno bazo. Ta je implementirana s pomočjo skriptnega jezika PHP (angl. *(P)ersonal home page Hypertext Preprocessor* - orodja za osebno spletno stran). Tako smo dosegli abstrakcijo spodnjega nivoja hiearhije (v našem primeru podatkovna baza) navzgor. Uporabniški vmesnik izvaja komunikacijo s spletnim strežnikom preko `HTTP POST` in `GET` poizvedbami, zato ta ne ve, kaj je pod njim in od kje ta pridobiva podatke. PHP skripte pa s

podatkovno bazo komunicirajo preko funkcij, katerim se poda `SQL` ukaze in parametre, za le-te [13].

4.3.1 Pregled osnov PHP-ja

PHP je skriptni jezik, ki je uporabljen pri programiranju spletnih aplikacij. Za posredovanje podatkov na strežnik se uporabljajo POST ali GET poizvedbe. Do njih, znotraj PHP skript, dostopamo preko parametrov `$_POST` in `$_GET`. Skripte naše aplikacije po koncu procesiranja podatkov serializirajo rezultat v `JSON` (angl. *JavaScript Object Notation*) formatu [17].

$_POST in $_GET

Preko POST ali GET parametrov lahko dostopamo do podatkov, ki so bili poslani na server. POST se uporablja za pošiljanje podatkov za vnos v sistem, GET pa se uporablja, ko delamo poizvedbe na spletni strežnik. Parametrom, znotraj PHP skript, samo podamo ime vrednosti, do katere želimo dostopati; da preverimo, ali je parameter znotraj poizvedbe podan, pa uporabimo funkcijo `isset` [16].

```
if (isset($_GET["qr_code"])) {
    $qr_code = $_GET['qr_code'];
}
```

JSON

JSON je format serializacije podatkov, kjer so le-ti zastavljeni v parih *ime*:*vrednost*. Je prostorsko optimizirana alternativa XML-u (angl. EXtensible Markup Language), kar pa v spletnih aplikacijah pomeni hitrejšo komunikacijo prek spleta.

Primer podatkov uporabnika aplikacije, seraliziranega v JSON formatu:

```
{
  "u_id": "igorP"
  "first_name": "Igor",
  "last_name": "Plavsic"
```

```
}
```

V JSON format se, v PHP-ju, serializira podatke z uporabo metode `json_encode`. Metodi podamo objekt, ta pa serializira objekt v zgoraj naveden rezultat [16].

```
$user["u_id"] = "igorP";
$user["first_name"] = "Igor";
$user["last_name"] = "Plavsic";
echo json_encode($user);
```

4.3.2 Konfiguracija povezave do podatkovne baze

Da s pomočjo PHP-ja lahko komuniciramo z bazo, moramo pred vsakim dostopom kreirati povezavo do MySQL-a ter nastaviti, s katero podatkovno bazo bomo upravljali. To dosežemo z metodama `mysql_connect`, ki ji podamo ime strežnika, uporabniško ime in geslo, za dostop do baze, druga pa je `mysql_select_db`, ki ji podamo ime podatkovne baze. Da te korake ne ponavljamo v vsaki skripti, ki potrebuje povezavo do baze, sta ta koraka ločena v svoji lastni skripti (*db_connect.php*), ki jo potem referenciramo v drugih skriptah [16].

```
// Povezava do MySQL-a
mysql_connect(DB_SERVER, DB_USER, DB_PASSWORD)
        or die(mysql_error());
// Izbira podatkovne baze
mysql_select_db(DB_DATABASE) or die(mysql_error())
        or die(mysql_error());
```

4.3.3 Poizvedbe na podatkovno bazo

Poizvedbe na podatkovno bazo se izvajajo preko metode `mysql_query`, ki ji podamo SQL (angl. *Structured Query Language*) stavek, kaj želimo storiti na bazi, ta pa vrne rezultat. Poizvedbe, ki se uporabljajo za potrebe aplikacije,

so: `SELECT` (poizvedba po vsebini tabel oz. večih tabel hkrati), `INSERT` (ukaz za vnos zapisa v tabelo), `UPDATE` (ukaz za ažuriranje obstoječega zapisa v tabeli) ter `DELETE` (uporabljen za izbris zapisa v tabeli). Vsaki od poizvedb lahko posredujemo tudi parametre, ki jih pospremimo s predponskim znakom $. Ali je poizvedba kaj vrnila, preverimo s klicem metode `mysql_num_rows`, ki ji podamo rezultat, ki ga je vrnil klic metode `mysql_query`. Pri poizvedbah, kot je SELECT, se moramo včasih sprehoditi čez več zapisov. To izvedemo z `while` zanko, skozi rezultat poizvedbe, kjer vsak zapis poizvedbe dobimo s klicem metode `mysql_fetch_array`. Primer uporabe metod je predstavljen v izseku kode [16].

```
// Primer klica izbrisa izdelka
$res1 = mysql_query("DELETE FROM products
        WHERE pid = $p_id")
// Poizvedba vseh izdelkov v bazi
$res2 = mysql_query("SELECT * FROM products")
if (mysql_num_rows($res1) > 0) {
        $response["success"] = 1;
}
while ($row = mysql_fetch_array($res2)) {
// procesiraj zapis
}
```

4.4 Uporabniški vmesnik

Vmesnik aplikacije sestavljajo pogledi, ki omogočajo interakcijo s podatki znotraj aplikacije. Uporabniški vmesnik oz. mobilna aplikacija ne vsebuje nobene poslovne logike, z izjemo klicev spletnega strežnika, ki dejansko upravlja s podatki.

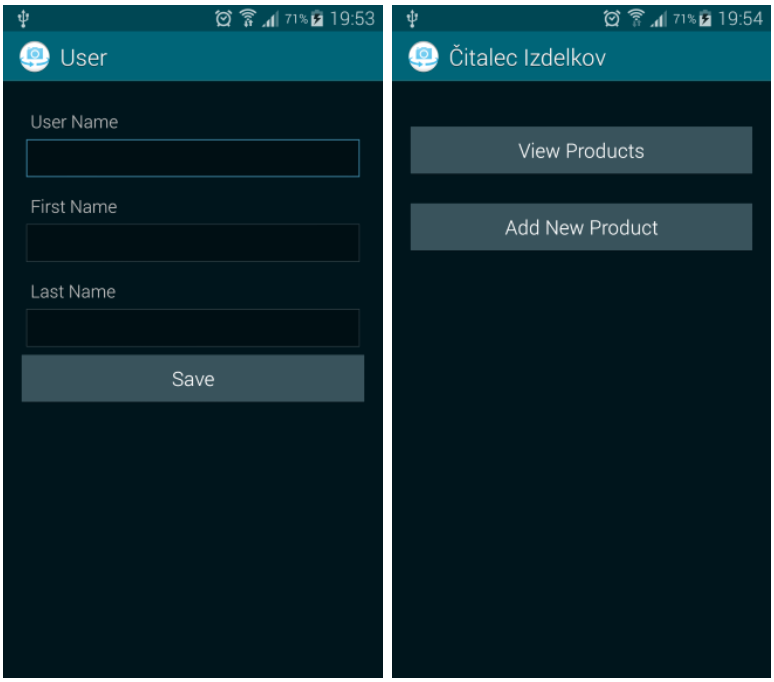
4.4.1 Opis pogledov v aplikaciji

Vpisni pogled

Prikaže se samo enkrat, in sicer ko prvič zaženemo aplikacijo. Uporabnik se shrani v bazo in aplikativne nastavitve aplikacije znotraj Andorid sistema. Podatke, ki jih vpišemo na tem pogledu, se uporabljajo ob kreiranju vseh drugih podatkov, s katerimi se srečujemo skozi delovanje aplikacije.

Glavni pogled

Po enkratnem vpisu v aplikacijo se prikaže glavni pogled. Vsebuje samo dva gumba; prvi omogoča vpogled vseh izdelkov, ki so shranjeni v bazi, drugi pa zažene proces odčitavanja izdelkov. Če odčitavanje vrne veljavno črtno kodo, se izvede poizvedba, ali izdelek že obstaja ter prikaže ustrezni naslednji pogled ali za vnos ali urejanje izdelkov.



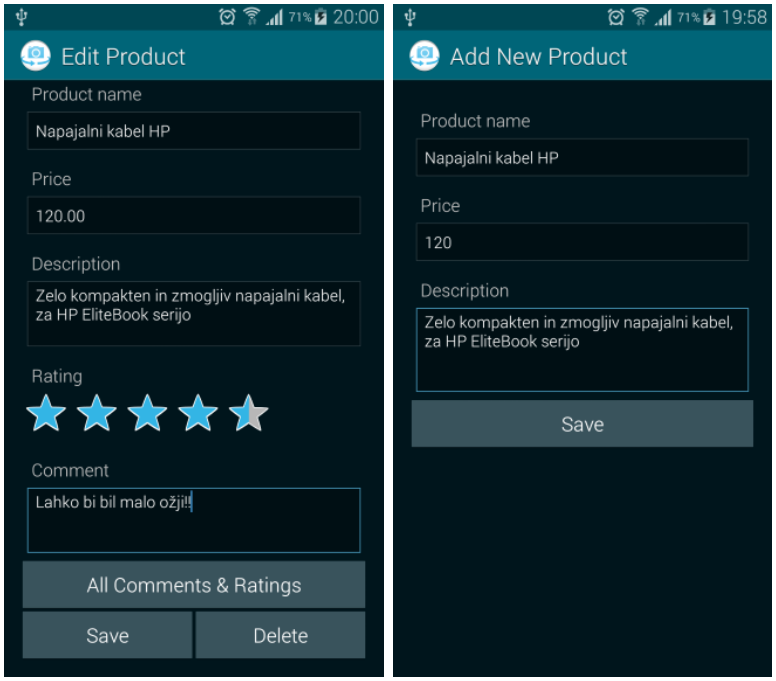
Slika 4.3: Vpisni in glavni pogled aplikacije

Pogled za kreiranje izdelka

Pogled se prikaže, če poizvedba z odčirano črtno kodo ne vrne rezultata. Pogled omogoča uporabniku vnos podatkov o izdelku, kot je naziv, cena in opis.

Pogled za urejanje izdelka

Se prikaže ob izbiri izdelka iz seznama izdelkov, ali ko odčitamo izdelek, ki že obstaja v sistemu. Pogled omogoča ažuriranje podatkov izdelka ali celo njegov izbris. Na tem pogledu je možen tudi vnos komentarja in ocene izdelka ter vpogled na vse komentarje in ocene drugih uporabnikov sistema.



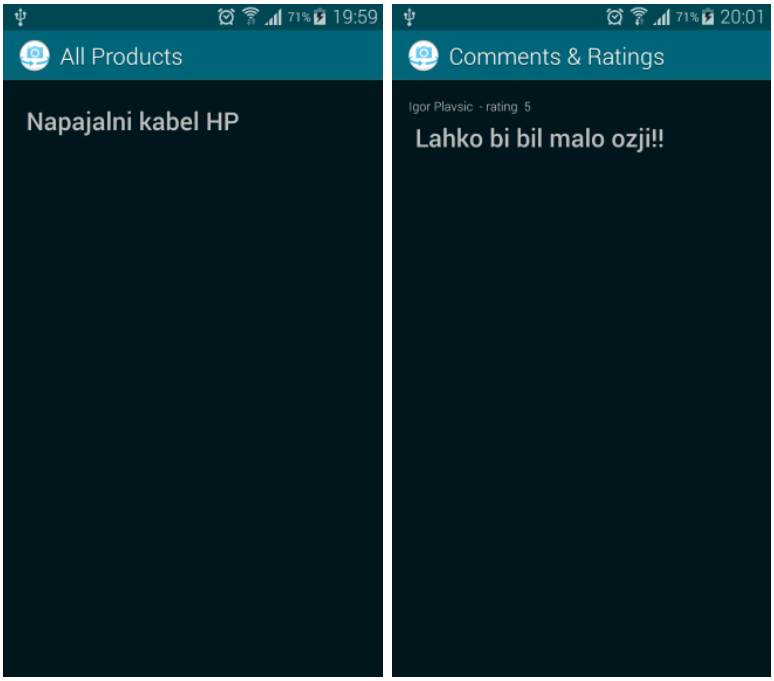
Slika 4.4: Pogleda za spremembo in vnos prebranega izdelka

Seznam vseh izdelkov

Pogled vsebuje seznam vseh izdelkov, shranjenih v sistemu. Do seznama dostopamo preko glavnega pogleda aplikacije. Če izberemo enega izmed izdelkov, se prikaže pogled za urejanje podatkov izdelka.

Pogled za prikaz komentarjev izdelkov

Pogled vsebuje seznam vseh ocen in komentarjev odčitanega oz. izbranega izdelka. Do njega dostopamo preko pogleda za urejanje podatkov izdelka.



Slika 4.5: Pogled seznamov izdelkov in komentarjev

Čitalnik črtnih kod

Je glavni pogled znotraj integriranega projekta, *Barcode Scanner*, ki odčitava črtne in QR kode, in se zaganja preko gumba na glavnem pogledu aplikacije. Rezultat njegove izvedbe je numerična predstavitev črtne kode, ki jo nato posredujemo v poizvedbo na spletni strežnik. Rezultat poizvedbe tudi določi, kateri pogled se bo prikazal naslednji.

Slika 4.6: Čitalec črtne kode

4.4.2 Opis komponent pogledov

Pogledi so sestavljeni iz večih komponent, ki so tudi same pogledi in omogočajo prikaz in interakcijo s podatki, s katerimi upravlja aplikacija:

TextView

Komponenta se uporablja za oznako vnosnih polj pogledov [18].

EditText

Vnosno polje preko katerega podajamo ali ažuriramo podatke, znotraj aplikacije. *EditText* omogoča tudi nastavitev vnosnega tipa, ki določi, kaj se lahko v polje vnaša, npr: ali je polje tekstovno ali numerično [19].

Button

Gumb je komponenta, ki uporabniku omogoča izvedbo akcij, ki vplivajo na delovanje aplikacije; v našem primeru so to zagoni poizvedb na spletni strežnik [20].

RatingBar

Ocenjevalna vrstica omogoča vizualno izbiro ocene. V našem primeru je to izbira zvezdic, v ozadju pa je vezano na numerično predstavitev vizualne izbire [21].

4.4.3 Upravljalniki razporeditve komponent

Pogledi v Androidu se uravnavajo glede na lastnosti upravljanikov razporeditve komponent. Sledijo opisi tistih, ki so uporabljeni v aplikaciji:

ListView

Vsebovalnik za seznam pogledov, ki jih definiramo v posebni XML datoteki, z razporeditvijo polj, ki se prikažejo na seznamu [22].

ScrollView

Ponavadi se uporablja na pogledih, ki vsebujejo veliko komponent. Če pogled presega velikost zaslona naprave, `ScrollView` obda pogled z drsniki, ki omogočajo premik pogleda na dele, ki so izven zaslona naprave [23].

LinearLayout

Je dinamični urejevalnik komponent, ki jih obdaja. Glede na velikost zaslona naprave razšija in krči velikost komponent, ki so na pogledu [24].

4.5 Komunikacija s strežnikom

Da je uporabniški vmesnik tekoč, komunikacija s stežnikom teče v asinhronih nalogah (angl. *AsyncTask*). Naloge nato preko posebej za aplikacijo kreiranega orodja `JSONParser` izvaja `HTTP` poizvedbe.

4.5.1 AsyncTask

Asinhrone naloge so namenjene uporabi pri poizvedbah na glavni niti uporabniškega vmesnika aplikacije. Naloge se izvajajo v ozadju, v ločeni niti, ko pa se izvede do konca, vrne rezultat na glavno nit uporabniškega vmesnika, kjer lahko podatke nato prikažemo v trenutnem pogledu [25].

Izvedba asinhrone naloge

Ko se asinhrona naloga izvede, naloga preide skozi štiri korake, ki jih je potrebno tudi pravilno implementirati.

onPreExecute()

Prvi korak se izvede pred izvedbo same naloge. Tukaj se lahko nastavi delovanje naloge glede na parametre, posredovane s pogleda ali za incializacijo in prikaz vrstice napredka.

doInBackground(Params...)

Izvedeno v zaledni niti takoj po `onPreExecute`. V tem koraku se izvajajo poizvedbe, podani pa so ji parametri naloge. Med izvajanjem metode se lahko kliče tudi `onProgressUpdate`, ki posodablja vrstico napredka. Rezultat metode je tudi vhodni parameter zadnjega koraka.

onProgressUpdate(Progress...)

Uporabljena za posodabljanje, kakršnih koli prikazovalnikov napredka naloge.

onPostExecute(Result)

V temu koraku se vrnemo na izvajanje na glavno nit uporabniškega vmesnika. Tu lahko posredujemo rezultat naloge na uporabniški vmesnik.

4.5.2 JSONParser

Za potrebe aplikacije smo ustvarili razred, ki izvaja `HTTP GET/POST` poizvedbe ter vrne rezultat le-teh, v obliki `JSONObject`a. Metodi `makeHttpRequest`, ki izvede poizvedbo, podamo URL do PHP skripte, ki jo želimo izvesti, vrsto poizvedbe GET ali POST, ter parametre, ki jih posredujemo v skripto.

Rezultat poizvedb je `JSONObject`, iz katerega deserializiramo podatke. Iz `JSONObject`a podatke pridobivano preko pomožnih metod, ki jim podamo ime vrednosti, ki jo želimo dobiti:

getString

Vrne vrednost niza, ki je mapirano na podano ime polja.

getInt

Vrne število, ki je mapirano na podano ime.

get...

`JSONObject` ima tudi implementirane metode za pridobivanje podatkov za druge primitivne podatkovne tipe. Med te spadajo tudi `boolean`, `double` in `long`.

getJSONArray

Vrne `JSONArray` (polje JSON objektov) mapirano na ime. Skozi polje se lahko sprehodimo in za pridobivanje zapisov s polja uporabimo metodo `getJSONObject`, ki ji podamo indeks zapisa. Nato nad osnovnim `JSONObject`om uporabljano ostale, prej navedene metode.

4.6 Integracija čitalca črtnih kod

Če želimo čitalca črtnih kod (angl. *Barcode Scanner*), ki je samostojna aplikacija, vključiti v našo, moramo naredi nekaj prilagoditev v našem projektu.

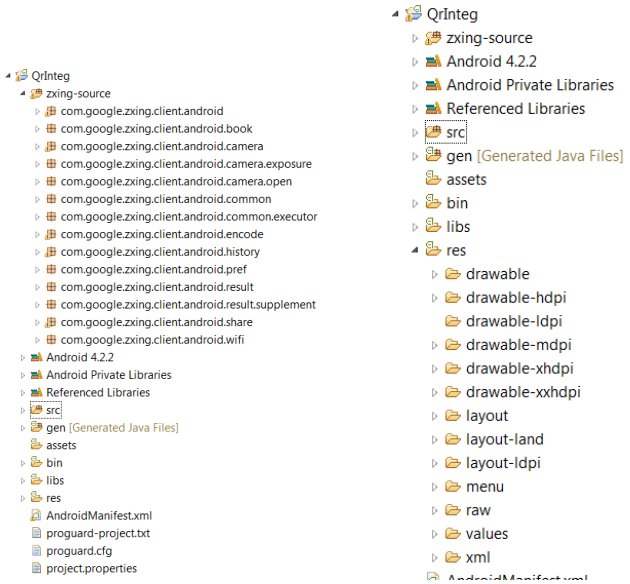
4.6.1 Prenos kode iz repozitorija

Kodo Googleovega odprtokodnega projekta pridobimo iz repozitorija kode GitHub (URL: *https://github.com/zxing/zxing*). To lahko storimo enostavno preko vtičnika za Eclipse za delo z GitHub portalom [26].

4.6.2 Združevanje projektov

Znotraj projekta lahko najdemo mapo `android`, ki vsebuje projekt ZXing, predelan za Android platformo. Vsebino tega projekta je potrebo združiti z našim. To dosežemo tako, da vso programsko kodo premaknemo v naš

projekt; za boljšo ažurnost projekta premaknemo vsebino v posebno datoteko, znotraj našega projetka (v našem primeru *zxing-source*). Za tem je potrebno združiti tudi vse resurse našega projekta, z ZXingovim. To dosežemo z združitvijo vsebin obeh `res/` map. Končni rezultat prikazuje Slika 4.7.



Slika 4.7: Rezultat združevanja projektov

Za konec pa moramo dopolniti še `AndroidManifest.xml` datoteko projekta tako, da dodamo aktivnost, ki zažene pogled za odčitavanje črtne kode, in sicer `CaptureActivity` integriranega projekta. Konfiguracijsko datoteko dopolnimo tudi z nastavitvami, ki jih čitalnik potrebuje, za pravilno delovanje.

```
<!-- Dodan CaptureActivity BarcodeScanner-ja, da lahko izvajam akcijo -->
<activity
    android:name="com.google.zxing.client.android.CaptureActivity"
    android:configChanges="orientation|keyboardHidden"
    android:label="ZXing"
    android:screenOrientation="landscape"
    android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
    android:windowSoftInputMode="stateAlwaysHidden" >
    <intent-filter>
        <action android:name="com.google.zxing.client.android.SCAN" />

        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

Slika 4.8: Dopolnjen AndroidManifest.xml

Poglavje 5

Sklepne ugotovitve

V diplomski nalogi smo razvili mobilno aplikacijo za odčitavanje in ocenjevanje izdelkov preko njihovih črtnih kod. Aplikacija za odčitavanje črtne kode uporablja rešitev implementirano v odprtokodnem projektu ZXing. Podatke pa, preko spletnega strežnika, shranjuje v oddaljeno podatkovno bazo.

Med razvojem aplikacije je prihajalo do veliko zanimivih zapletov, ki sem jih moral rešiti. Najbolj zanimiv se je pojavil pri združevanju projektov, ki ga po vsem prebrodenem, ne bi priporočal in bi raje šel v smeri tega, da bi naprava morala imeti nameščen drug čitalec črtnih kod, ki bi ga nato klical iz naše aplikacije. Probleme sem imel tudi pri skriptiranju PHP skript, ki bi se jim lahko izognil z neposredno povezavo na bazo, a bi tako žrtvoval "čisti"uporabniški vmesnik, ki bi tako moral vsebovati poslovno logiko.

Literatura

[1] Android Version History. (Povzeto 17.10.2014) Dostopno na: http://en.wikipedia.org/wiki/Android_version_history.

[2] Official Android Blog. (Povzeto 10.9.2014) Dostopno na: http://officialandroid.blogspot.com/.

[3] Android software development. (Povzeto 10.9.2014) Dostopno na: http://en.wikipedia.org/wiki/Android_software_development

[4] Android developer tools. (Povzeto 10.9.2014) Dostopno na: https://developer.android.com/tools/help/index.html

[5] View. (Povzeto 10.9.2014) Dostopno na: http://developer.android.com/reference/android/view/View.html

[6] View. (Povzeto 10.9.2014) Dostopno na: http://developer.android.com/reference/android/view/ViewGroup.html

[7] Activity. (Povzeto 10.9.2014) Dostopno na: http://developer.android.com/guide/components/activities.html

[8] Intent. (Povzeto 10.9.2014) Dostopno na: http://developer.android.com/reference/android/content/Intent.html

[9] PreferenceManager. (Povzeto 10.9.2014) Dostopno na: http://developer.android.com/reference/android/preference/PreferenceManager.html

[10] What is a QR code. (Povzeto 10.9.2014) Dostopno na:
http://www.qrcode.com/en/about/

[11] ZXing project. (Povzeto 10.9.2014) Dostopno na:
https://github.com/zxing/zxing

[12] XAMPP Installer. (Povzeto 10.9.2014) Dostopno na:
https://www.apachefriends.org/index.html

[13] About the Apache HTTP Server Project (Povzeto 12.9.2014) Dostopno na:
http://httpd.apache.org/ABOUT_APACHE.html

[14] MySQL (Povzeto 12.9.2014) Dostopno na:
http://en.wikipedia.org/wiki/MySQL

[15] Relational database (Povzeto 12.9.2014) Dostopno na:
http://en.wikipedia.org/wiki/Relational_database

[16] PHP Manual (Povzeto 12.9.2014) Dostopno na:
http://si1.php.net/manual/en/

[17] PHP (Povzeto 12.9.2014) Dostopno na:
http://en.wikipedia.org/wiki/PHP

[18] TextView (Povzeto 16.9.2014) Dostopno na:
http://developer.android.com/reference/android/widget/TextView.html

[19] EditText (Povzeto 16.9.2014) Dostopno na:
http://developer.android.com/reference/android/widget/EditText.html

[20] Button (Povzeto 16.9.2014) Dostopno na:
http://developer.android.com/reference/android/widget/Button.html

[21] RatingBar (Povzeto 16.9.2014) Dostopno na:
http://developer.android.com/reference/android/widget/RatingBar.html

[22] ListView (Povzeto 16.9.2014) Dostopno na:
http://developer.android.com/reference/android/widget/ListView.html

[23] ScrollView (Povzeto 16.9.2014) Dostopno na:
http://developer.android.com/reference/android/widget/ScrollView.html

[24] LinearLayout (Povzeto 16.9.2014) Dostopno na:
http://developer.android.com/reference/android/widget/LinearLayout.html

[25] AsyncTask (Povzeto 16.9.2014) Dostopno na:
http://developer.android.com/reference/android/os/AsyncTask.html

[26] EGit (Povzeto 16.9.2014) Dostopno na:
http://www.eclipse.org/egit/